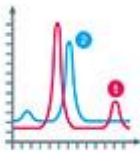


NHẬP MÔN LẬP TRÌNH KHOA HỌC DỮ LIỆU

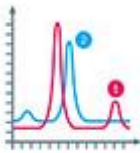
Bài 5: Ngôn Ngữ Lập Trình Python (4)

Nhắc lại kiến thức bài trước



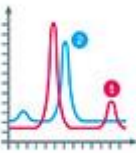
- **Kiểu dữ liệu tuần tự**: là kiểu dữ liệu cho phép xử lý dữ liệu bằng cách xử lý từng-phần-tử-con-một
- **Danh sách (list)**: dãy các phần tử, khai báo bên trong cặp ngoặc vuông, nội dung có thể thay đổi
- **Hàng (tuple)**: dãy các phần tử, khai báo bên trong cặp ngoặc tròn, nội dung cố định (không thay đổi)
- **Range (miền)**: có thể xem như một dạng tuple đặc biệt gồm các số nguyên, chuyên dùng cho lặp for
- **Chuỗi (str)**: một dạng tuple đặc biệt gồm nhiều chuỗi có độ dài 1 ký tự

Nhắc lại kiến thức bài trước

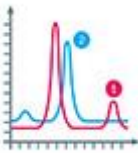


- Các kiểu dữ liệu này có chung đặc điểm:
 - Bản chất là các đối tượng, được viết một cách tự nhiên
 - Rất nhiều phương thức hỗ trợ việc xử lý
 - Sử dụng chung 2 hệ thống chỉ mục (âm và dương)
 - Sử dụng chung kỹ thuật cắt lát (bằng chỉ mục)
 - Sử dụng chung 3 phép toán: +, *, in
- Chuỗi có rất nhiều kỹ thuật định dạng nội dung
- List và Tuple có thể được tạo bằng comprehension
- Nhiều hàm dựng sẵn (built-in) xử lý các kiểu dữ liệu này: len, max, min, all, any, filter, sorted, sum, zip,...

Nội dung

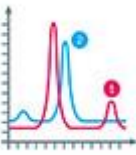


1. Set (tập hợp) và Frozenset (tập hợp tĩnh)
2. Dictionary (từ điển)
3. Module và Package
4. Bài tập



Phần 1

Set (tập hợp) và Frozenset (tập hợp tĩnh)



Giới thiệu và khởi tạo

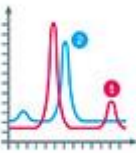
- Set = tập hợp các đối tượng (không trùng nhau)
- Khai báo trực tiếp bằng cách liệt kê các phần tử con đặt trong cặp ngoặc nhọn ({}), ngăn cách bởi phẩy

```
>>> basket = {'apple', 'orange', 'apple', 'pear'}  
>>> print(basket)  
{'orange', 'pear', 'apple'}      # xóa trùng nhau
```

- Tạo set bằng constructor

```
s1 = set([1, 2, 3, 4])      # {1, 2, 3, 4}  
s2 = set((1, 1, 1))        # {1}  
s3 = s1 - s2                # {2, 3, 4}  
s4 = set(range(1,100))     # {1, 2, 3,..., 98, 99}
```

Khởi tạo



- Tạo set bằng set comprehension

```
# a = {'r', 'd'}
```

```
a = {x for x in 'abracadabra' if x not in 'abc'}
```

- Set không thể chứa những đối tượng mutable (có thể bị thay đổi), mặc dù chính set lại có thể thay đổi

```
a = set(([1,2], [2,3])) # lỗi
```

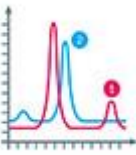
```
a = set((1,2), (2,3)) # {(1, 2), (2, 3)}
```

```
a.add("abc") # {(1, 2), "abc", (2, 3)}
```

- Frozenset giống set, nhưng không thể bị thay đổi

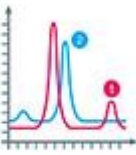
```
b = frozenset((1,2), (2,3)) # {(1, 2), (2, 3)}
```

```
b.add("abc") # lỗi
```



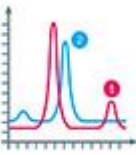
Các phép toán trên set

```
a = set('abracadabra')      # {'d', 'r', 'c', 'b', 'a'}
b = set('alacazam')        # {'z', 'c', 'm', 'l', 'a'}
# Phép Hiệu: thuộc a nhưng không thuộc b
print(a - b)               # {'r', 'd', 'b'}
# Phép Hợp: thuộc a hoặc b
# {'a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'}
print(a | b)
# Phép Giao: thuộc cả a và b
print(a & b)               # {'a', 'c'}
# Phép Xor: thuộc hoặc a, hoặc b nhưng không phải cả 2
# {'r', 'd', 'b', 'm', 'z', 'l'}
print(a ^ b)
```

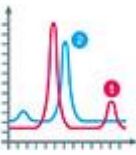
Các phương thức của set

- Một số phương thức thường hay sử dụng
 - `add(e)`: thêm e vào tập hợp
 - `clear()`: xóa mọi phần tử trong tập hợp
 - `copy()`: tạo một bản sao của tập hợp
 - `difference(x)`: tương đương với phép trừ đi x
 - `difference_update(x)`: loại bỏ những phần tử trong x khỏi tập
 - `discard(e)`: bỏ e khỏi tập
 - `remove(e)`: bỏ e khỏi tập, báo lỗi nếu không tìm thấy e
 - `union(x)`: tương đương với phép hợp với x
 - `intersection(x)`: tương đương với phép giao với x



Các phương thức của set

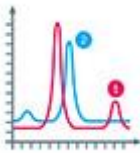
- Một số phương thức thường hay sử dụng
 - `isdisjoint(x)`: trả về True nếu tập không có phần chung nào với x
 - `issubset(x)`: trả về True nếu tập là con của x, tương đương với phép so sánh $\leq x$
 - `issuperset(x)`: trả về True nếu x là tập con của tập, tương đương với phép so sánh $\geq x$
 - `pop()`: lấy một phần tử ra khỏi tập (không biết trước)
 - `symmetric_difference(x)`: tương đương với phép $\wedge x$



Phần 2

Dictionary (từ điển)

Dictionary (từ điển)



- Từ điển là một danh sách các từ (key) và định nghĩa của nó (value)
 - Yêu cầu các key không được trùng nhau, như vậy có thể xem từ điển như một loại set

- Từ điển có thể khai báo theo cú pháp của set

```
>>> dic = {1:'one', 2:'two', 3:'three'}
```

```
>>> print(dic[1])
```

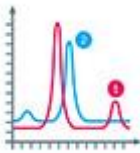
```
'one'
```

```
>>> dic[4]='four'
```

```
>>> print(dic)
```

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

Dictionary (từ điển)



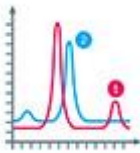
- Chú ý: chỉ những loại dữ liệu immutable (không thể thay đổi) mới có thể dùng làm key của từ điển

```
dic = { (1,2,3):"abc", 3.1415:"abc" }
```

```
dic = { [1,2,3]:"abc" } # lỗi
```

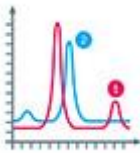
- Một số phép toán / phương thức thường dùng
 - `len(d)`: trả về độ dài của từ điển (số cặp key-value)
 - `del d[k]`: xóa key k (và value tương ứng)
 - `k in d`: trả về True nếu có key k trong từ điển
 - `k not in d`: trả về True nếu không có key k trong từ điển
 - `pop(k)`: trả về value tương ứng với k và xóa cặp này đi
 - `popitem()`: trả về (và xóa) một cặp (key, value) tùy ý

Dictionary (từ điển)



- Một số phép toán / phương thức thường dùng
 - `get(k)`: lấy về value tương ứng với key k
 - Khác phép `[]` ở chỗ `get` trả về `None` nếu `k` không phải là key
 - `update(w)`: ghép các nội dung từ từ điển `w` vào từ điển hiện tại (nếu key trùng thì lấy value từ `w`)
 - `items()`: trả về list các cặp (key, value)
 - `keys()`: trả về các key của từ điển
 - `values()`: trả về các value của từ điển
 - `pop(k)`: trả về value tương ứng với `k` và xóa cặp này đi
 - `popitem()`: trả về (và xóa) một cặp (key, value) tùy ý

Dictionary (từ điển)



- Dùng zip để ghép 2 list thành từ điển

```
>>> l1 = ["a", "b", "c"]
```

```
>>> l2 = [1, 2, 3]
```

```
>>> c = zip(l1, l2)
```

```
>>> for i in c:
```

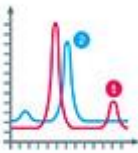
```
...     print(i)
```

```
...
```

```
('a', 1)
```

```
('b', 2)
```

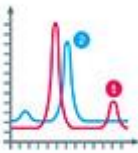
```
('c', 3)
```



Phần 3

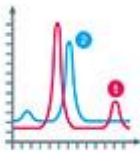
Module và Package

Module



- Một file mã nguồn trong python được xem là một module
 - Có phần mở rộng .py
 - Mọi hàm, biến, kiểu trong file là các thành phần của module
- Sử dụng module:
 - Khai báo import module đó: `import <tên-module>`
 - Có thể khai báo import cùng lúc nhiều module cách nhau bởi dấu phẩy
 - Nếu muốn sử dụng các hàm, biến trong module thì cần viết tường minh tên module đó
 - Hoặc có thể import riêng một hàm hoặc nhiều hàm, cú pháp: `from <tên-module> import fuc1, fuc2,..., fucN`

Package



- Package = Thư mục các module (lưu trữ vật lý)

```
import numpy
```

```
A = array([1, 2, 3]) # lỗi
```

```
A = numpy.array([1, 2, 3]) # ok
```

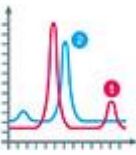
```
import numpy as np
```

```
B = np.array([1, 2, 3]) # ok
```

```
from numpy import array
```

```
C = array([1, 2, 3]) # ok
```

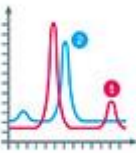
- Module và Package giúp quản lý tốt hơn mã nguồn
- Gom, nhóm các hàm, biến, lớp xử lý cùng một chủ đề, giúp phân cấp và sử dụng dễ dàng hơn



Phần 4

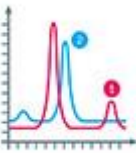
Bài tập

Bài tập



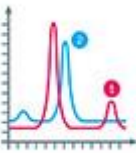
1. Tạo một tập hợp gồm các phần tử từ 0 đến 99, in chúng ra màn hình
2. Tạo một tập hợp gồm các số nguyên lẻ trong khoảng từ 1 đến 199, in chúng ra màn hình
3. Tạo một tập hợp gồm các số nhập vào từ bàn phím (nhập trên 1 dòng, cách nhau bởi ký tự trống), tìm và in ra số phần tử của tập, giá trị lớn nhất và nhỏ nhất trong tập
4. Cho D là từ điển định nghĩa cách đọc các chữ số ở tiếng Anh, hãy in ra các value của D theo thứ tự tăng dần

Bài tập



5. Nhập một từ điển D, hãy in ra các value khác nhau trong từ điển
6. Nhập một từ điển D có các value là các số nguyên, hãy in ra màn hình 3 giá trị value lớn nhất
7. Nhập một string S, hãy tạo từ điển D trong đó key là các chữ xuất hiện trong S còn value tương ứng là số lần xuất hiện các chữ đó trong S
 - Ví dụ: S = “dai hoc thuy loi”
 D = { ‘d’:1, ‘a’:1, ‘i’:2, ‘ ’:3, ‘h’:2,
 ‘o’:2, ‘c’:1, ‘t’:1, ‘u’:1, ‘y’:1, ‘l’:1 }

Bài chữa



1. Tập từ 0 đến 99

```
num_set = set(range(0,100))
for n in num_set:
    print(n)
```

2. Tập các số lẻ từ 1 đến 199

```
num_set = set(i for i in range(0,200) if i%2==1)
print(num_set)
```

3. Tập nhập từ bàn phím

```
num_set = set(int(i) for i in input("Cac so: ").split())
print("SPT:", len(num_set))
print("Max:", max(num_set))
print("Min:", min(num_set))
```